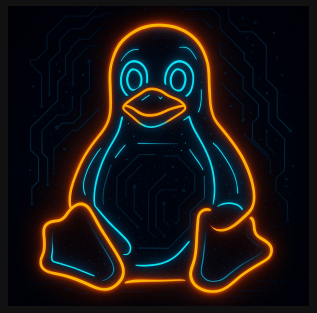

Linux Forensics Cheat Sheet (Debian/Ubuntu/Kali)

Nikolaos Kranidiotis • osec.gr

Quick Triage

- `w` – Show who's logged in (current users) and their sessions.
- `last -n 10` – List last 10 logins (from `/var/log/wtmp`) and `lastb` for failed attempts.
- `ps -aux --forest` – List processes with tree view to spot odd parent-child relations. Look for unusual processes or ones running as root with high PIDs.
- `ss -tunap` (or `netstat -tunap`) – Show all listening ports and active network connections with associated processes.
- `find / -mtime -1 -type f 2>/dev/null` – Find files modified in the last 24 hours (adjust timeframe) to spot recent changes.
- `find / -perm -4000 2>/dev/null` – Identify SUID binaries (potential privilege escalation tools or planted backdoors).
- `ls -alR /proc/*/exe 2>/dev/null | grep deleted` – Check for running processes with deleted executables (indicative of malware in memory).
- Quickly check for log tampering: e.g. `ls -l /var/log/*` for zero-length files (logs wiped by attacker).

Suspicious Artifacts Locations

- **Temporary files:** Check ephemeral dirs like `/tmp`, `/var/tmp`, `/dev/shm` for dropped payloads or scripts.
- **Spool & Run:** Inspect `/var/spool` (e.g. mail, cron) and `/var/run` for unexpected files or folders.
- **User directories:** Look in user home directories for hidden files (prefix `.`) or unusual binaries; e.g. `~/.ssh/`, `~/.cache/`, `~/.config/`.

- **System binaries:** Verify integrity of critical system files in `/bin`, `/usr/bin`, `/sbin` (attackers may replace common binaries). Use `dpkg -V` or `debsums -c` on Debian to find altered files.
- **Libraries & Modules:** Scan `/lib`, `/usr/lib` for rogue .so files or kernel modules. Attackers might drop rootkits in `/lib/modules/$(uname -r)` or via `/etc/ld.so.preload`.
- **Web directories:** If a web server is present, check `/var/www` (or web app directories) for webshells or unfamiliar files.

Persistence & Autostart

- **Systemd Services:** List autostart services with `systemctl list-unit-files --state=enabled` and inspect `/etc/systemd/system/` (and subdirs like `multi-user.target.wants/`) for unauthorized .service files. Examine service definitions using `systemctl cat name.service` for suspicious commands.
- **Cron Jobs:** Check system cron schedules in `/etc/crontab`, `/etc/cron.d/`, and user crontabs in `/var/spool/cron/crontabs/`. Run `crontab -l -u <user>` for each user. Look for @reboot entries or strange schedules. Also review `/etc/anacrontab` and `/var/spool/anacron` for persistent tasks. Use `atq` to list pending at jobs.
- **Init Scripts:** Investigate legacy init locations: `/etc/init.d/` and scripts linked in `/etc/rc*.d/` (SysV init). Also check `/etc/rc.local` (executed at end of boot).
- **Shell Profiles:** Attackers may persist via login scripts. Inspect `/etc/profile`, `/etc/bash.bashrc`, and scripts in `/etc/profile.d/` (system-wide). For each user, review `~/.bashrc`, `~/.bash_profile`, `~/.profile`, and even desktop autostart files (e.g. `~/.config/autostart/*.desktop`) for injected commands.
- **Kernel Modules:** Review `/etc/modules` (modules to load at boot) and `/etc/modprobe.d/` configs for suspicious entries. Use `lsmod` to see loaded modules and compare against known baseline. Attackers may load rootkit modules or alter module configs.
- **Other Mechanisms:** Check `/etc/inetd.conf` or `/etc/xinetd.d/` for unauthorized network services. Also look at environment triggers like `~/.ssh/authorized_keys` with embedded command= options (backdoor access) and `~/.ssh/rc` scripts.
- **Scheduled Tasks (systemd timers):** Run `systemctl list-timers --all` to list scheduled systemd timers (similar to cron) which might execute persistent malware on schedule.

Logs & Event History

- **Authentication Logs:** `/var/log/auth.log` (Debian/Ubuntu) or `/var/log/secure` (RHEL) – record SSH logins, sudo use, and auth failures.

Search for keywords like "Accepted password", "Failed password", "sudo" etc. Also check `/var/log/faillog` for failed login stats.

- **System Logs:** `/var/log/syslog` (Debian) or `/var/log/messages` – general system messages, daemon outputs, and kernel events. Review for unusual errors or service restarts around incident time.
- **Kernel & Device Logs:** `/var/log/kern.log` – kernel-specific logs (watch for module load messages or Oops). `/var/log/dmesg` – boot and driver messages (USB device insertions will appear here or in syslog). Use `dmesg | grep -i usb` to find USB events (external device connections).
- **Login Records:** `/var/log/wtmp` (all logins) and `/var/log/btmp` (failed logins) store user login history. Use `last` and `lastb` to read these. `/var/log/lastlog` shows last login per user (use `lastlog` command).
- **Application Logs:** Look into service-specific logs under `/var/log/` (e.g. `apache2/access.log`, `apache2/error.log`, `mysql/error.log`, etc.) for signs of exploitation or exfiltration. Also check `/var/log/cron` for cron job executions and `/var/log/daemon.log` for other daemon events.
- **Audit Logs:** If Auditd is running, inspect `/var/log/audit/audit.log` for detailed security events (e.g. file access, process execution). Use `ausearch` and `aureport` to query these logs.
- **Log Integrity:** Look for gaps or anomalies in timestamps that could indicate log tampering. Logs wiped by attackers (e.g. suddenly no entries for a period) or log files with suspiciously new creation dates are red flags.

File Carving & Deleted Files

- **Foremost:** Carve files from disk images by file type signatures. Example: `foremost -t jpg,pdf -i disk.img -o ./recovery` (carves JPG and PDF files from `disk.img` into `./recovery`). Omit `-t` to use all default file types or edit `foremost.conf` to specify types.
- **Scalpel:** A faster Foremost variant requiring configuration of file types in `/etc/scalpel/scalpel.conf`. After uncommenting desired file types, run `scalpel /dev/sda1 -o /path/to/recovery_dir` to carve deleted files from a device or image.
- **extundelete:** Undelete files on ext3/ext4 filesystems using journal data. Usage: `extundelete --restore-all /dev/sdX1` (or `--restore-file=path` for specific files) to recover to `RECOVERED_FILES` directory. Ensure the volume is unmounted (or mounted read-only) before running.
- **The Sleuth Kit (TSK):** Use TSK for low-level analysis:
  - `fls -r -a -m / image.dd > bodyfile.txt` – List all files (including deleted) from image and save in timeline bodyfile.
  - `ils -m image.dd` – List metadata of deleted files (in bodyfile format).
  - `icat image.dd <inode>` – Extract file content by inode number (even if deleted, if not overwritten).
- **Photorec:** (Part of TestDisk) Scans block device or image for known file signatures (similar to Foremost). Useful for carving common file types via an interactive console UI.
- Always work on disk *images* or read-only mounts. For live systems, consider `dd` imaging the volume or using `ddrescue` for damaged disks, then perform carving on the image to avoid altering the source.

Memory Acquisition & Analysis

- **LiME (Linux Memory Extractor):** Loadable kernel module for memory dumps. Compile the module on a system with identical kernel. Usage: `insmod lime-$(uname -r).ko "path=/dump/path.mem format=lime"` to write a full RAM dump (use `format=raw` for raw format). LiME also supports output over TCP (e.g., `"path=tcp:[COLLECTOR_IP]:4444"` to send dump to a remote listener).
- **AVML:** Microsoft's memory acquisition tool (precompiled) – simply run `./avml memory.lime` to dump memory to a file. Produces LiME format output without needing a kernel module (works on Azure VMs and Linux systems).
- **Volatility Framework:** Use Volatility 3 (or Volatility 2 with profile) to analyze Linux memory dumps. Common plugins include:
  - `linux.pslist` – List processes in memory.
  - `linux.pstree` – Process tree, showing parent/child relationships.
  - `linux.lsmod` – List loaded kernel modules (detect unknown modules/rootkits).

- `linux.ifconfig`/`linux.netstat` (Volatility 2) or `linux.sockstat` – Show network interfaces and connections in memory.
- `linux.bash` – Recover bash command history from memory (even if wiped on disk).
- `linux.malfind` – Detect hidden or injected code segments in process memory.

Example: `vol.py -f memory.lime linux.pslist` (Volatility 3 uses `vol` or `vol3` command). Ensure you have the correct symbol table (Volatility 3 can use `banners.Banners` to identify kernel version and require a generated JSON profile).

- **Rekall:** An alternative memory analysis framework (fork of Volatility) with built-in Linux profiles. Usage is similar, e.g., `rekall -f memory.lime pslist`.
- **Memory analysis tips:** Look for suspicious processes not seen on disk, network connections in memory, signs of credential material (e.g. passwords in process memory), and kernel hooks. Consider dumping suspicious process memory (Volatility's `linux.proc.*` plugins or using `dd` on `/proc/[pid]/mem` with proper permissions) for further malware analysis.

Timeline Reconstruction

- **The Sleuth Kit & mactime:** Generate filesystem timeline from an image:
  - Use `fls -r -m / image.dd > bodyfile.txt` to create a body file of all FS entries (with MAC times).
  - Then `mactime -b bodyfile.txt -d -z UTC > timeline.csv` to produce a CSV timeline. (Use appropriate timezone for the system.)

  The timeline will include file creations, modifications, and access times to correlate system events.
- **log2timeline/Plaso:** Use `log2timeline.py` to create a supertimeline from various log files and artifacts. Example: `log2timeline.py incident.plaso disk.img` will parse file system and known log formats into a Plaso storage file. Then use `psort.py -o CSV -w timeline.csv incident.plaso` to output a combined timeline. Plaso incorporates many artifact parsers (event logs, browser history, etc.) beyond just file timestamps.
- **Manual Log Timeline:** Correlate system logs (auth, syslog, etc.) around key timestamps. For instance, if an intrusion was detected at a certain time, review all log entries 10–15 minutes before and after across auth.log, syslog, web logs, etc., to piece together a sequence of events.
- **Timeline Analysis:** Look for clusters of events (multiple file modifications or log entries in short succession) that might indicate attacker activity. Use tools like Timesketch or ELK stack to visualize timelines for large datasets, or a spreadsheet to filter and sort CSV timelines.
- **MAC times caveat:** Remember that file timestamps can be altered by attackers (anti-forensics) or reset by certain activities (e.g. copying files preserves timestamps by default). Always consider the timeline in context with other evidence (such as logs or memory captures).

Network & Exfiltration

- **Live Connections:** Use `ss -tulnp` or `netstat -tulnp` to list listening ports and established TCP/UDP connections with PID/Program names. Check for unexpected listeners (e.g., backdoor shells on high ports) or outbound connections to unfamiliar IPs. Also `lsof -i -P -n` can reveal processes with network sockets open.
- **Network Sniffing:** Capture traffic with `tcpdump` (e.g., `tcpdump -nn -w capture.pcap -i eth0` to log all interface traffic) or `tshark` for more filtering. If exfiltration is suspected, filter by known attacker IP or uncommon ports/protocols (e.g., DNS or ICMP tunneling).
- **Protocol Analysis:** Use `tshark -r capture.pcap -Y "http or dns or ftp"` to quickly find cleartext creds or data leaks in common protocols. Look at DNS queries for suspicious domains, HTTP for large data posts or strange user-agents, etc.
- **Zeek (Bro):** Run Zeek on a pcap (`zeek -r capture.pcap`) to generate high-level logs of connections, HTTP sessions, DNS queries, files transferred, etc. Review these logs (conn.log, http.log, dns.log, files.log) for anomalies like large outbound transfers or connections to blacklisted domains.
- **System Logs for Exfil:** Check shell history or commands for use of network utilities (`scp`, `ftp`, `wget`/`curl` to external sites, `netcat`) around the incident. Also inspect `/var/log/auth.log` for port-forwarding or tunneling attempts via SSH (e.g., using `ssh -D` or `ssh -L` options).
- **Firewall Logs:** If available (e.g., iptables logging or UFW logs in `/var/log/ufw.log`), review for unusual outbound connections or data volume. Commands like `iptables -L -v -n` show packet counts per rule, which might indicate volumes of traffic if logging rules are in place.
- **Indicators of Exfil:** Large spikes in network usage, presence of archive files (zip, 7z, tar) in temp directories, or use of transfer tools like `rsync`, `scp` by suspicious accounts. Also search for any encoded blobs in logs or on disk that might represent packaged data (e.g., a big Base64 blob in an outbound email or script).

Malware Hunting & IOCs

- **Filesystem Search:** Hunt for common malicious patterns. For instance:
  - Search for Base64 or other encoding usage: `grep -R "base64" /etc /home 2>/dev/null` (attackers often hide commands or data in base64).
  - Search for suspicious keywords: `grep -R -E "nc -e|/bin/sh|bash -i|exec 5<>/dev/tcp" / 2>/dev/null` to find signs of reverse shells/ backdoors.
  - Find scripts with odd shebangs or paths (e.g., referencing `/dev/shm` or `/tmp`).
- **Hash & Scan:** Generate hashes (MD5/SHA1/SHA256) of unidentified binaries and compare against malware databases. Use `sha256sum *` to list hashes, then cross-check any unknown ones via VirusTotal or threat intelligence sources.
- **Binary Clues:** Use `strings` on suspicious binaries for hints (IP addresses, URLs, suspicious function names). Check for packer markers

(e.g., "UPX" in output). Use `file` to identify file type (ELF, script, etc.) and `ldd` on ELF to see linked libraries (malware may link unusual libraries).

- **Rootkit Checks:** Look for rootkit behaviors:
  - Hidden modules: cross-verify `lsmod` vs `cat /proc/modules` for discrepancies. Run tools like `chkrootkit` and `rkhunter` (with updated DB) to detect known rootkits.
  - LD Preload: Check `/etc/ld.so.preload` for any entries (normally empty/nonexistent; any .so listed is suspicious).
  - Hidden processes: Compare `ps` output with `proc` filesystem entries or use `pstree` to spot processes with no parent or odd names. Also ensure no unexpected process is listening on network ports (per above network checks).
- **Autorun IOCs:** Re-check persistence locations for malware clues:
  - New or modified service unit files in `/etc/systemd/system/` (especially with unusual names).
  - Unexpected executables in `/etc/cron.*` or `/etc/init.d/`.
  - Suspicious additions in shell profiles (e.g., an alias or function that launches malware, or a LD_PRELOAD export).
- **User Activity:** Review shell history files (e.g. `~/.bash_history`) for strange commands (downloading files, altering permissions, compiling code). Even if attackers cleared history, partial traces might remain or be found in memory (Volatility's linux.bash plugin). Also check `~/.ssh/authorized_keys` for unfamiliar keys (a planted key = persistence).
- **Indicators in Logs:** Look for multiple failed logins followed by a success (brute-force) in auth logs, or a sudden use of `sudo` by a new user (privilege escalation). Check for references to known hack tools or exploit scripts in logs. Search logs for keywords like "Accepted publickey" (new SSH key usage), "UID=0" (unexpected root actions), or known malware process names.
- **YARA Scan:** Consider running YARA rules on suspicious files or memory dumps to identify known malware. Use community rule sets (e.g., from AlienVault OTX or GitHub repos) for Linux threats. Scan critical dirs and compare hits against threat intel databases.

Credential & Sensitive Data

- **Password Hashes:** `/etc/shadow` contains hashed passwords for user accounts (root-only access). Check for new accounts or weak hashes (passwordless entries are marked `!` or `*`). `/etc/passwd` lists all users; ensure no unauthorized accounts, especially none with UID 0 (root) besides root itself.
- **SSH Keys:** Inspect `~/.ssh/authorized_keys` for unauthorized public keys (attackers often add their own key for persistence). Also collect any private keys (e.g., `~/.ssh/id_rsa`) on the system – they might be stolen. Check `~/.ssh/known_hosts` for hosts the attacker connected to or from.
- **Stored Credentials:** Browser-stored passwords (if a desktop system): For Firefox, check `~/.mozilla/firefox/` (look for `logins.json` and `key4.db` files); for Chrome/Chromium, `~/.config/google-chrome/Default/Login Data` (SQLite DB). Also consider saved VPN passwords, email

client creds, or Wi-Fi passwords (NetworkManager configs).

- **Keyrings & Vaults:** Linux desktops use keyrings (e.g., GNOME Keyring files in `~/.local/share/keyrings/` or `~/.gnome2/keyrings/`) and KDE Wallet. These may contain saved credentials (SSH passphrases, web passwords). If obtainable (unlocked or cracked), they can reveal lots of stored secrets.
- **Cloud/API Keys:** Look for cloud config files containing keys or tokens: e.g., `~/.aws/credentials` (AWS keys), `~/.azure/azureProfile.json`, `~/.gcp/`, or config files for CI/CD tools (Jenkins, Docker registries). These can be high-value for attackers.
- **Memory for Creds:** In memory dumps, search for cleartext credentials: e.g., strings that look like passwords or AWS secrets. Also consider dumping process memory of authentication daemons (ssh, sudo) via volatility to see if any passwords or keys lingered. If kernel dumps (core dumps in `/var/crash`) exist from crashes, they might contain secrets in process memory at crash time.

Hidden & Temp Files

- **Dotfiles & Hidden Dirs:** Use `find / -type f -name ".*"` and similarly for `-type d` to locate hidden files/directories. Attackers hide data or scripts with a dot prefix. Common examples: hidden cron scripts (e.g. `/etc/cron.d/.something`), or directories named ".. " (dot-dot-space) to confuse listing. Inspect any such hidden artifact closely.
- **/tmp & /dev/shm:** World-writable temp locations are often used for malicious payloads or staging. Look for executable files in `/tmp` and `/dev/shm` (e.g., `find /tmp -type f -perm -111`). Also list any FIFOs (named pipes) via `find /tmp -type p` – malware may use these for communication. Remember these paths are not persisted after reboot, so live response is key.
- **/dev abuse:** Review the `/dev` directory for any regular files. All entries should be device nodes or symlinks. An attacker might hide data as a "device", e.g., create `/dev/.sock` or `/dev/ptmx\n` (with a newline) to conceal it. Use `find /dev -type f` to identify any non-device files.
- **Extended Attributes:** Linux files can have extended attributes (xattr) which might store hidden data or flags. Use `lsattr -R /` to find files with attributes like `i` (immutable) which attackers use to prevent log deletion or tampering. Use `getfattr -R -d /` to dump all xattrs and search for suspicious content or unusually large xattrs.
- **Cleanup Traces:** Attackers often remove their files. Look for clues of deleted files: log entries complaining about missing files, bash history showing use of `rm` commands, or directory gaps (forensic tools like extundelete or TSK fls can help enumerate deleted files). Also check `/var/tmp` and `/var/crash` for any leftovers or crash dumps.
- **Memory-only Files:** Some malware runs purely in memory (fileless). However, they might leave handles open. Check `/proc/pid/exe` symlinks: if an executable shows "`(deleted)`", that process might be running a file that was deleted (common for fileless malware). Also examine `/proc/pid/fd` for any mapped files in deleted state or living in /dev/shm.